Distributed Systems

COMP90015 2019 Semester 1
Tutorial 06

Questions

Q1. What is a thread and life cycle of a thread?

Q2. Compare worker pool multi-threading architecture with the thread-per-request architecture?

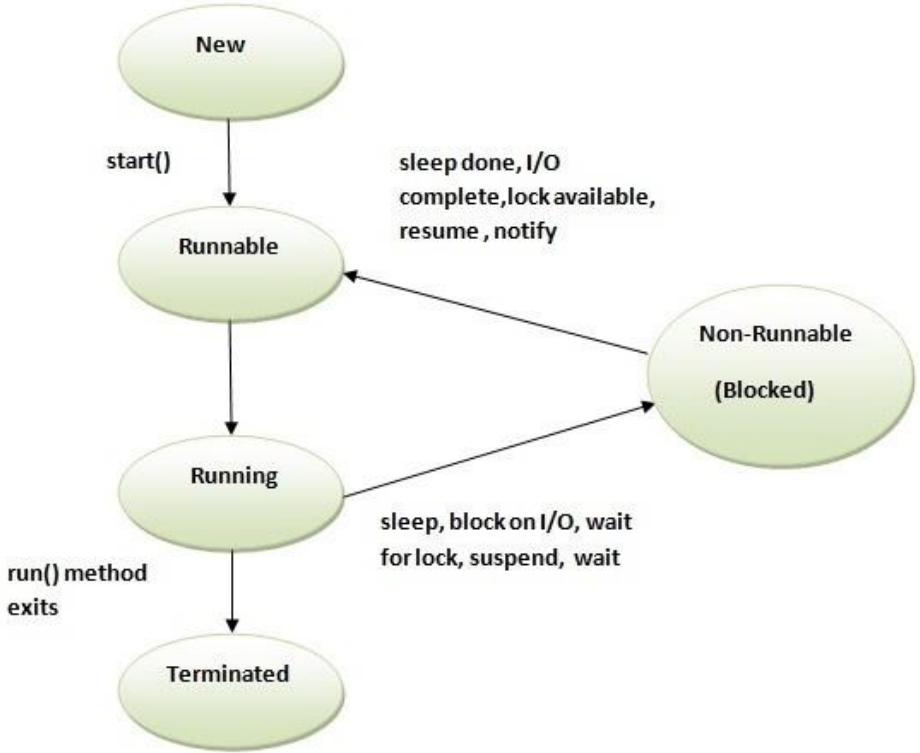
Q3. Please explain Remote procedure call (RPC) and describe its key components.

Q4. Explain the steps involved in Java RMI to build a distributed system.

Q1. What is a thread and life cycle of a thread?

- A Thread is a piece of code that runs in concurrent with other threads.
- Each thread is a statically ordered sequence of instructions.
- Threads are used to express concurrency on both single and multiprocessors machines.
- Threads share the same address space and therefore can share both data and code

Thread Lifecycle



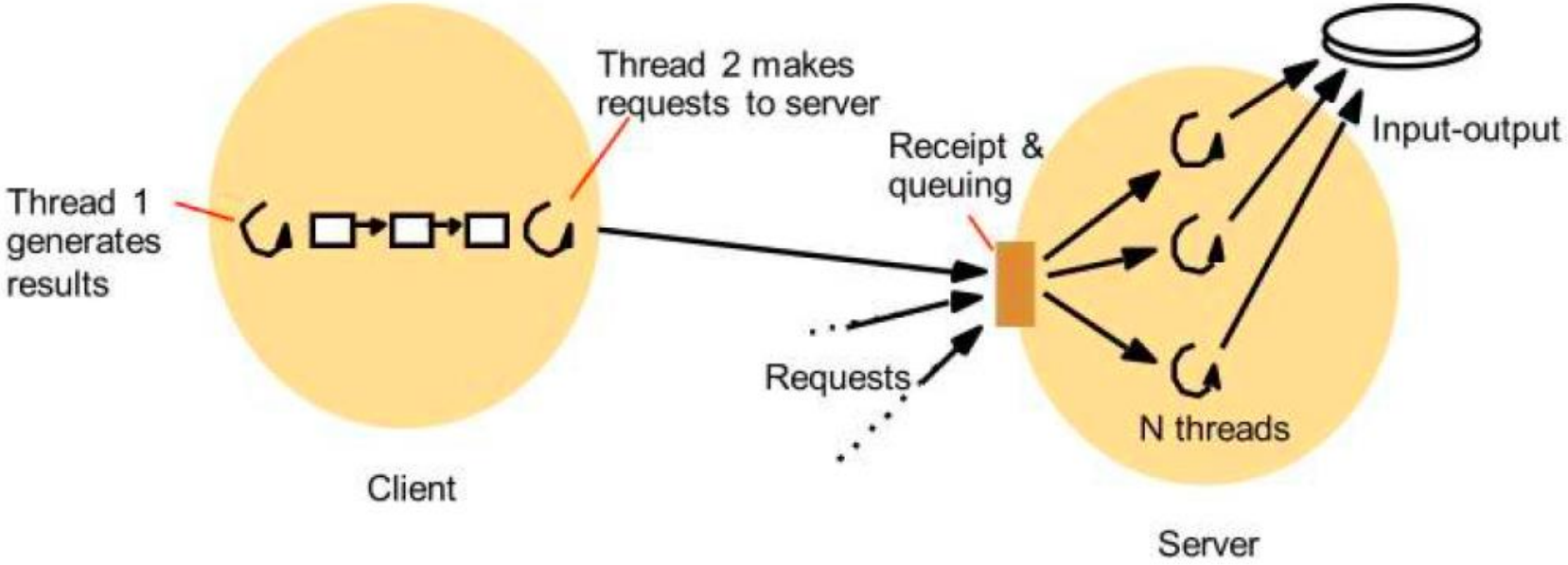
Q2. Compare the worker pool multi-threading architecture with the thread-per-request architecture.

Worker pool architecture

The server creates a fixed number of threads called a worker pool. As requests arrive at the server, they are put into a queue by the I/O thread and from there assigned to the next available worker thread.

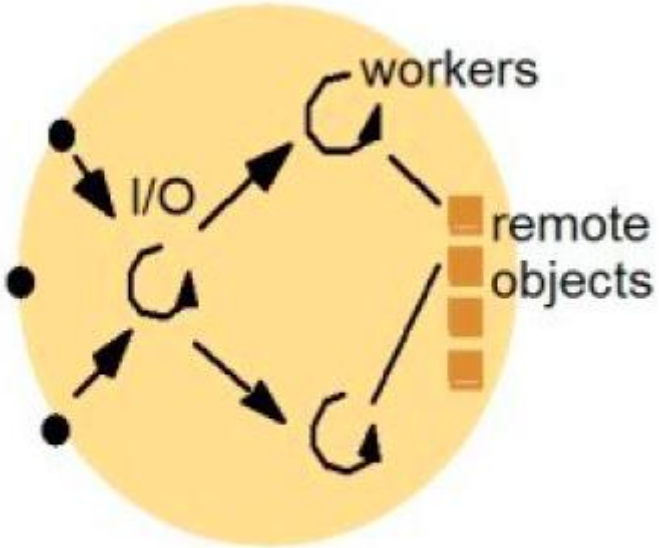
Server creates worker pool ➜ request comes in, put into a queue ➜ assigned to an available worker thread.

Worker pool architecture



Thread-per-request architecture

Thread created for each request, when the request is finished, the thread is deallocated.



a. Thread-per-request

Remote Procedure Call

Q3. Please explain Remote procedure call (RPC) and describe its key components.

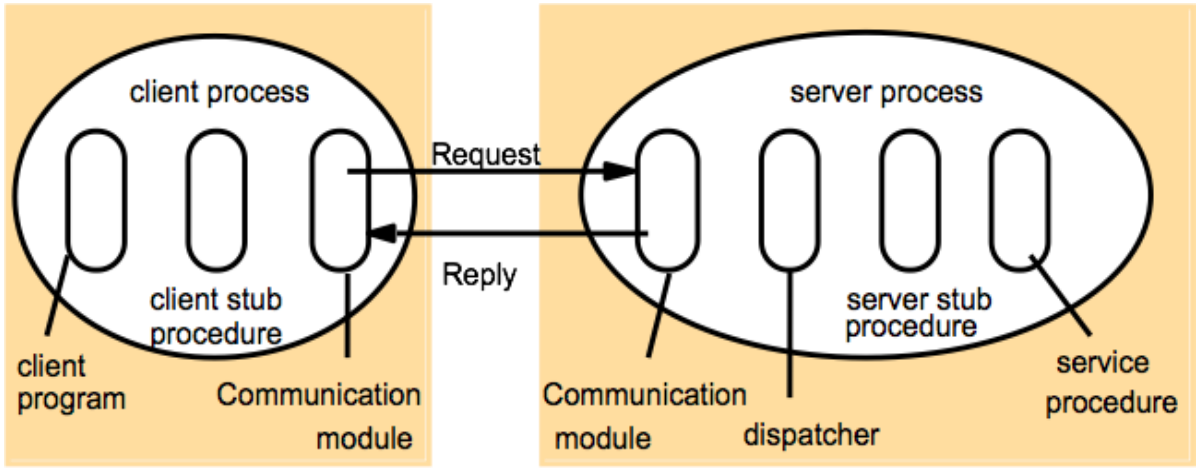
- RPCs enable clients to execute procedures in server processes based on a defined service interface.

- Used in procedural languages such as Fortran, C, and GO.

Remote Procedure Call (RPC) and key components

Key components of RPC:

- Communication Module
- Client Stub Procedure
- Dispatcher
- Server stub procedure

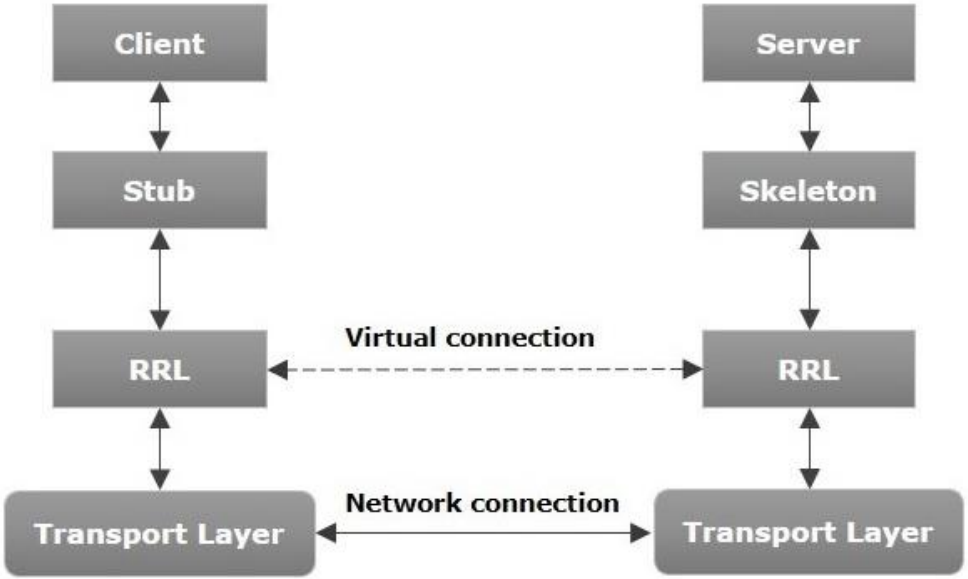


Remote procedure call (RPC) and key components

- **Communication Module**
  Implements the desired design choices in terms of retransmission of requests, dealing with duplicates and retransmission of results
- **Client Stub Procedure**
  Behaves like a local procedure to the client. Marshals the procedure identifiers and arguments which is handed to the communication module
  Unmarshalls the results in the reply
- **Dispatcher**
  Selects the server stub based on the procedure identifier and forwards the request to the server stub
- **Server Stub Procedure**
  Unmarshalls the arguments in the request message and forwards it to the Service Procedure. Marshalls the arguments in the result message and returns it to the client

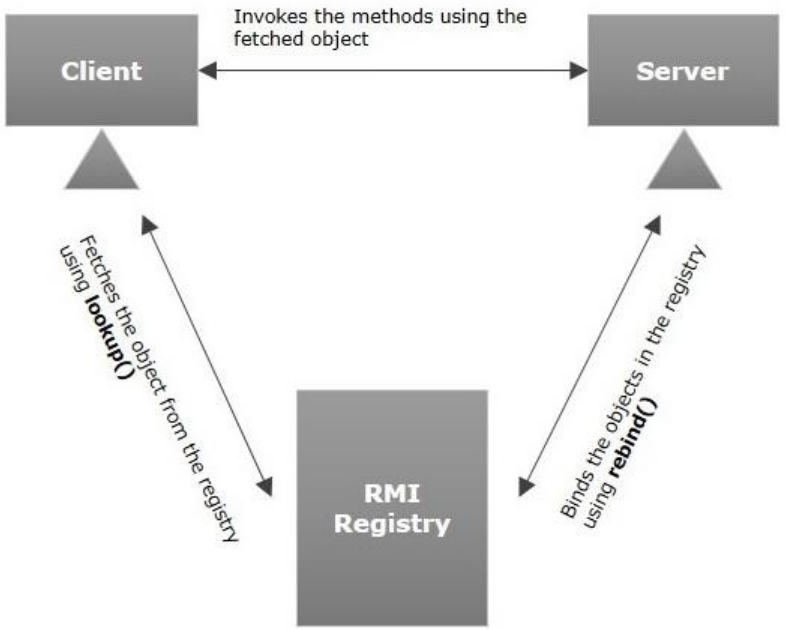
Java RMI

Q4. Explain the steps involved in Java RMI to build a distributed system.

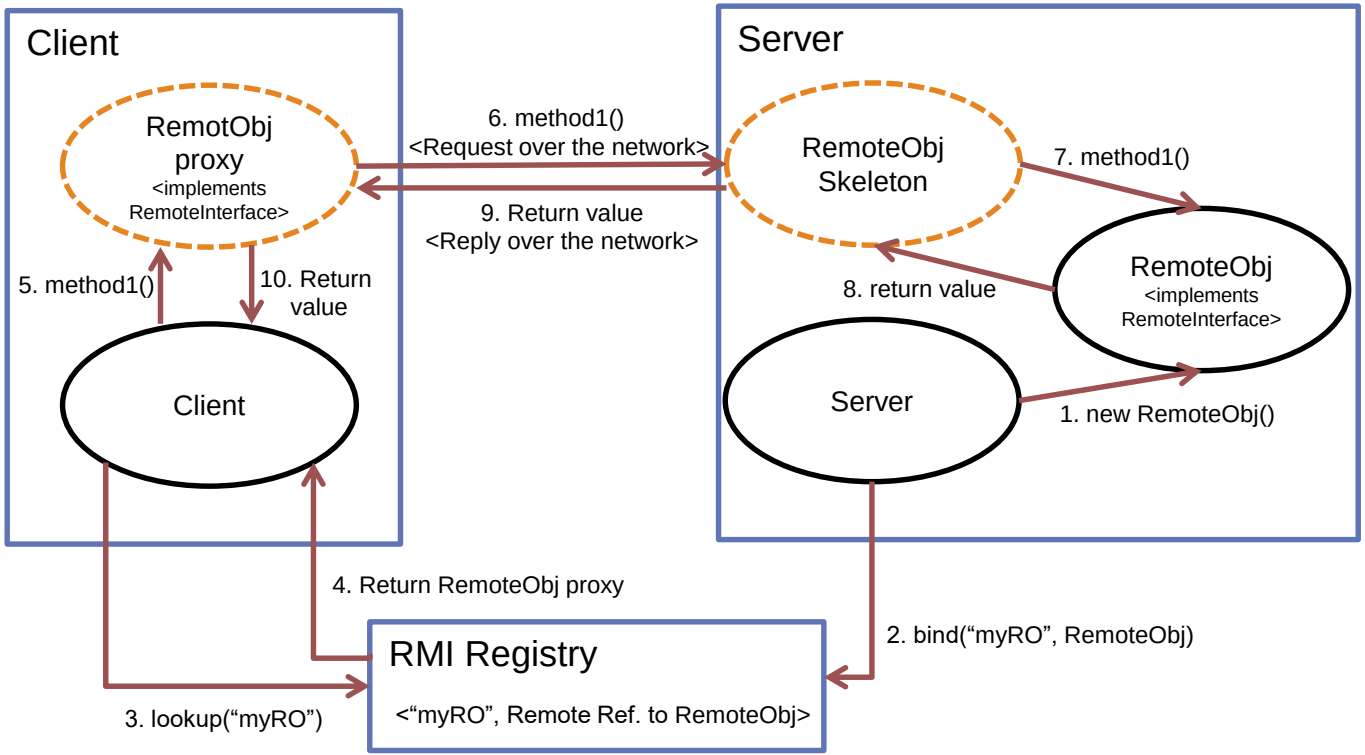


Java RMI

- Explain the steps involved in Java RMI to build a distributed system.



Java RMI Overview



Explain the steps involved in Java RMI to build a distributed system.

- Define the remote interface
  - Defines the methods that can be remotely invoked
  - Extends java.rmi.Remote interface
  - All methods throw java.rmi.RemoteException
  - Needs to be included in the client and in the server programs
- Server program
  - Create the remote object
  - Implements the remote interface
  - Extends UnicastRemoteObject
  - Implement the actual server
    - Export the remote object into the Java RMI runtime so that it can receive remote calls
    - Locate the RMIregistry and publish the remote object on it (the registry stores <name, remoteObjRef> pairs.
- Client program
  - Locate the RMIregistry and lookup the remote object by name
  - Invoke methods on the remote object as if it was a local one

Demo Time!

RMI Demo